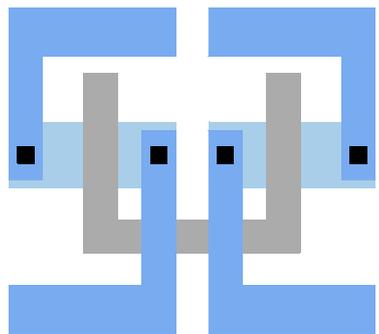




# Vorlesung „Tools“ gnuplot



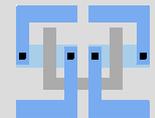
Schaltungstechnik  
und Simulation

Dr. Michael Ritzert

[michael.ritzert@ziti.uni-heidelberg.de](mailto:michael.ritzert@ziti.uni-heidelberg.de)

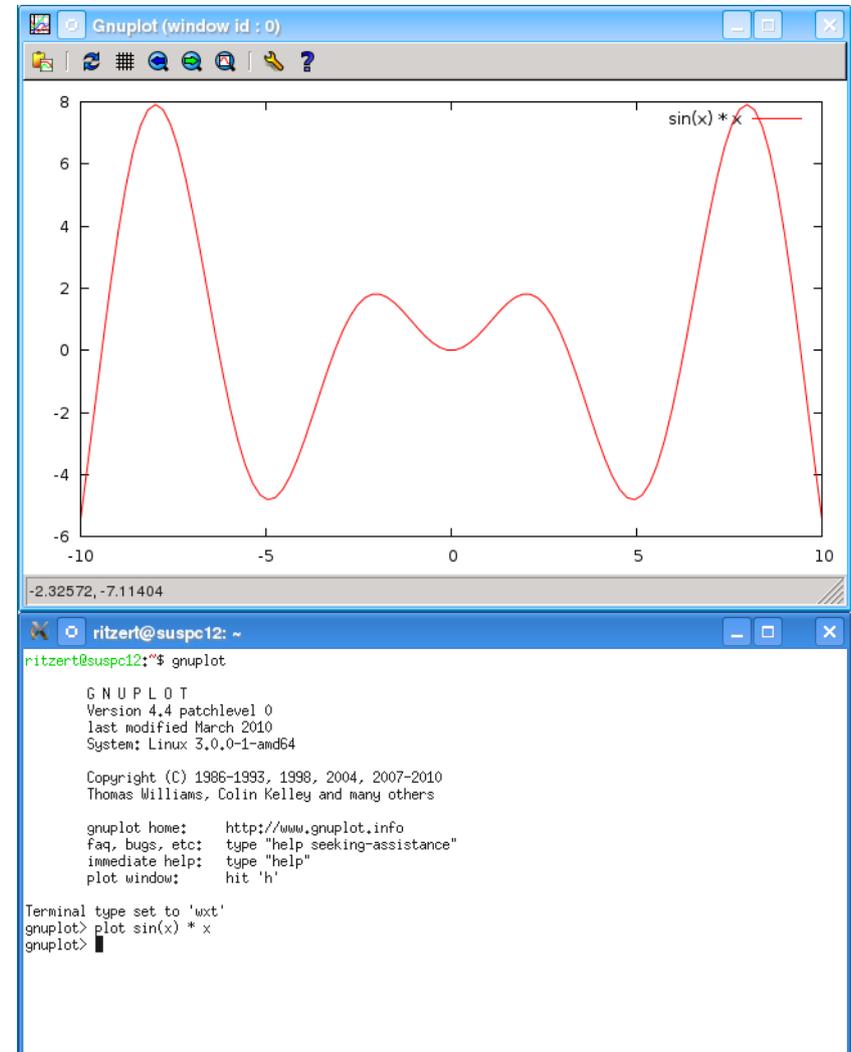
21.06.2023

- Tool zum 2- und 3-dimensionalen Plotten von Daten: Funktionen und Datenfiles.
- Einfache Fits von Funktionen
- Viele Ausgabeformate:
  - Auf Bildschirm,
  - Postscript,
  - Bitmap (JPEG, PNG, GIF, ...),
  - EPS + LaTeX (sehr nützlich für Papers, etc.),
  - EMF (für Powerpoint-Präsentationen)
  - Noch VIEL mehr...
- Hauptsächlich Steuerung über Kommandozeile.  
Details zu Befehlen immer mit `help Befehl`.  
Fast alle Befehle können viel mehr als hier gezeigt...
- <http://www.gnuplot.info>  
Galerie: <http://gnuplot.sourceforge.net/demo/>



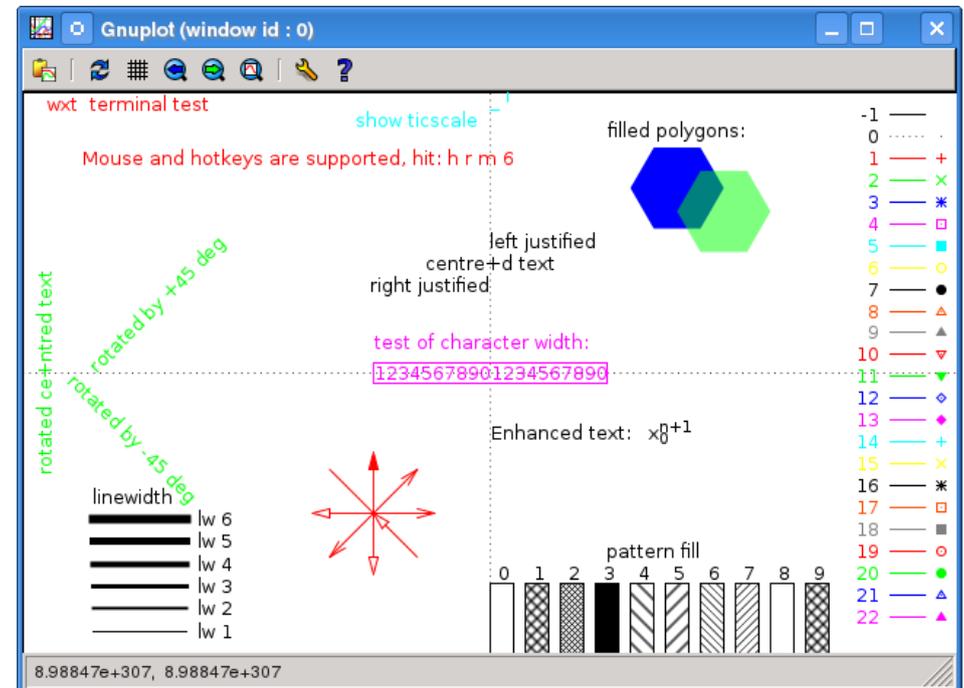
# gnuplot : erste Plots

- gnuplot starten mit  
> gnuplot
- beenden mit  
> exit
- Einfache 2D-Plots:  
> plot *Funktion*
- „Übliche“ Funktionen sofort verfügbar. Potenz  $x^y$  mit  $x^{**}y$ .
- Y-Achse skaliert per default automatisch, x-Achse nur bei Plots aus Daten-Files.
- Achsen skalieren:  
> set xrange [-20:20]  
ebenso yrange, zrange, cbrange
- log. Achse: set logscale y
- Mehrere Plots:  
ein plot-Befehl, Funktionen mit Komma getrennt.
- Kurzformen fast aller Befehle.



# gnuplot : Plotstile

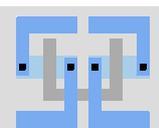
- Angabe mit „with“ nach der zu plottenden Funktion.
- Linien, Punkte, Flächen.  
lines, points, filledcurve
- Treppen, Histogramme, Fehlerbalken.  
steps, boxes, histogram, errorbars
- Farben  
linecolor rgb '*name*'  
(ohne with)
- Eigene Stile (Kombinationen von Eigenschaften)  
definieren mit set style.



- Üblich: Text-File als Input (binary möglich)
- Beliebige Anzahl Spalten, mit Leerzeichen getrennt  
Adressierung über `using`.
- Kürzeste Form:  
`plot 'Datei' (with lines title "Titel")`
- Mehrere Datensätze pro Datei mit zwei Leerzeilen getrennt.  
Adressierung über `index`.
- Kommentare mit `#` am Zeilenanfang.
- DEMO
  - Datenfile plotten, verschiedene Spalten und Blöcke
  - Plot-Stile
  - 2. y-Achse
- Besondere Dateinamen:
  - `' '`: gleiche Datei wie zuvor
  - `' - '`: lies von Kommandozeile. Ende mit `e`.

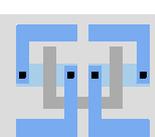
```
1 1 1
2 4 8
3 9 27
4 16 64
```

```
1 1
2 1.41
3 1.73
4 2
```



- gnuplot kann nach Belieben mit Werten rechnen, auch aus mehreren Spalten.
- Die Syntax ist `using` mit `()` um das Argument, das berechnet werden soll. Die Spalten werden mit `$1`, `$2`, ... angesprochen.
- `$0` ist ein Sonderfall: Nummer des aktuellen Punkts (ab 0).
- `plot 'Datei' using 3:($1+sqrt($2)) with lines`
- Funktionen können auch vorher definiert werden:  
`squaredSum(a,b)=sqrt(a**2+b**2)`
- Ternärer Operator: `?:`  
`a ? b : c`  $\Rightarrow$  `b`, falls die Bedingung `a` wahr ist, `c` sonst.  
Häufig verwendet mit NaN („not a number“  $\Rightarrow$  ignorieren):  
`plot '...' using 1:($2>7 ? $3 : NaN)`  
 $\Rightarrow$  wie `plot '...' using 1:3`, überspringt aber Zeilen mit `$2<=7`.

- Überschrift über dem Plot:  
`set title "Text"`
- Legende:  
`set key on/off` zum ein-/ausschalten  
`set key bottom left` oder `set key at x,y`
- Legendeneintrag für einen Plot:  
`plot ... title "Text"`
- Plot in Legende nicht auflisten:  
`plot ... notitle`
- Achsenbeschriftung:  
`set xlabel "Text"`. Ebenso für `y`, `cb`, etc.
- Striche an Achsen:  
(un)set `xtics`, etc. zum ein-/ausschalten  
`set xtics rotate winkel` zum Drehen der Beschriftung
- Gitter im Plot:  
`set grid [x|y]`

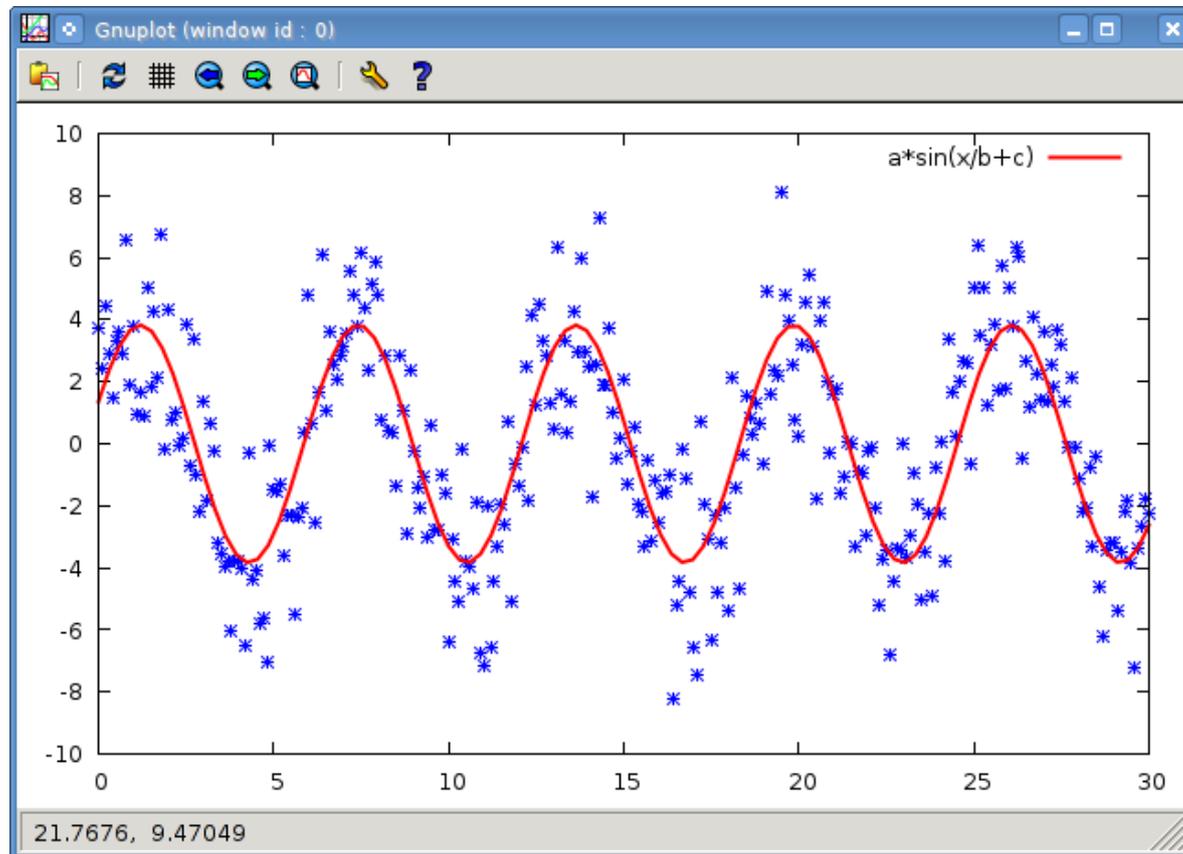


- Textlabels im Plot:  
`set label "Text" at x,y`
- Pfeile im Plot:  
`set arrow from x1,y1 to x2,y2`
- Linie = Pfeil ohne Kopf: `set arrow ... nohead`  
Pfeile in beide Richtungen: `heads`
- Beschriftung der x-Achse aus Datenfile:  
`plot ... using x:y:xtic(spalte)`
- y-Achse bei x=0:  
`set yzeroaxis`
- Änderungen übernehmen (letzten `plot`-Befehl mit neuen Einstellungen ausführen):  
`replot`

- String für Labels, etc. dynamisch erzeugen:  
`sprintf` und `gprintf`.
- Eingabe: String mit Formatanweisungen und einzufügende Werte.
- `sprintf`: „normale“ C-Formatanweisungen.
- `gprintf`: gnuplot-spezifische Formatanweisungen,  
aber nur 1 Argument!  
s. `help format specifiers`
- ```
gnuplot> a=3.7
gnuplot> s=sprintf( "%.1f**2 = %.3f", a, a**2 )
gnuplot> print s
3.7**2 = 13.690
```
- Format der Achsenbeschriftung ändern:  
`set format x "%.3f"` (gnuplot-Anweisungen)  
z.B. `"%.1s %c"` für Zahlen mit 1 Nachkommastelle und SI-Präfix.

# „Fits“

- Gegeben: Messpunktreihe und Modellfunktion mit freien Parametern
- Gesucht: Belegung der Parameter, so dass die Funktion bestmöglich zu den Messpunkten passt.
  - „bestmöglich“ => Maß nötig. Üblich: Summe der quadratischen Abweichungen zwischen Funktion und Datenpunkten („least squares“).



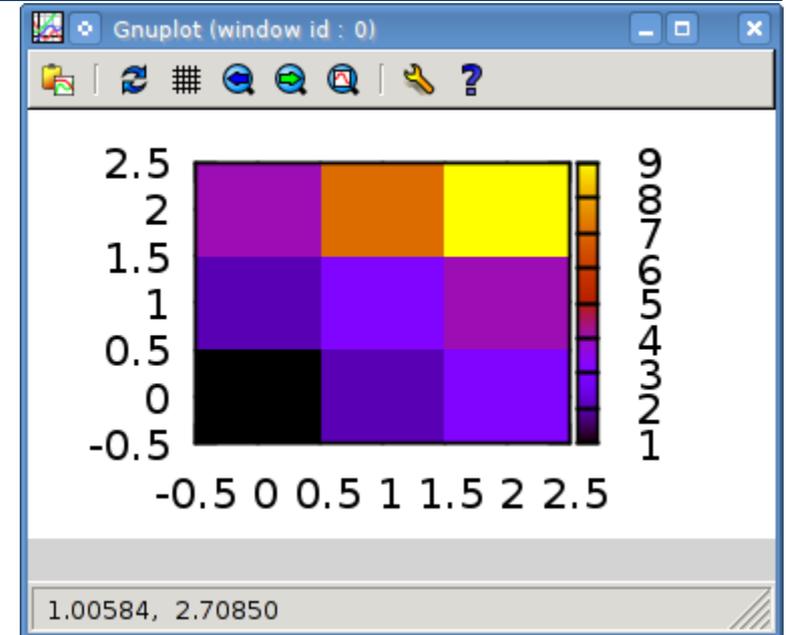
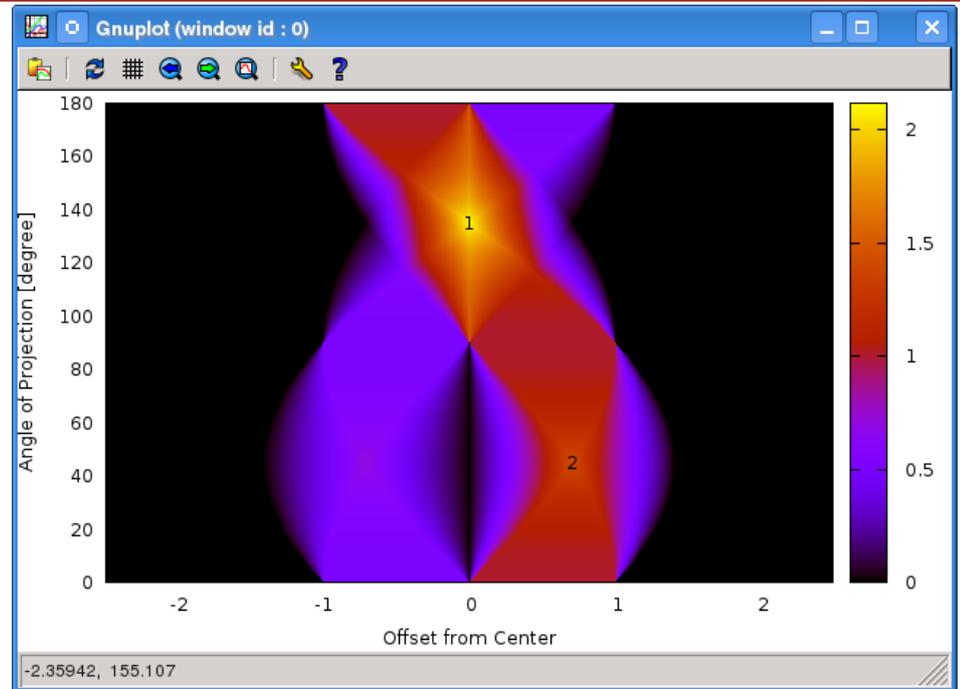
- `fit Funktion 'Datensatz' via var1, var2, ...`
- Anfangswerte für Variablen: einfach vorher zuweisen  
`var1=42`
- Nach dem Fit: Variablen entsprechen dem Fit-Ergebnis.  
⇒ können im Title, Labels, etc. eingebaut werden
- Zu fittenden Bereich einschränken:  
`fit [von:bis] ...`
- (Erlaubte Werte für Variablen können nicht einfach eingeschränkt werden.)
- Gewichte können aus einer 3. Spalte gelesen werden.  
einfach `using` um 1 Spalte erweitern.
- Probleme bei sehr großen oder kleinen Zahlen!  
(groß/klein:  $>\sim 10^{10}$ ,  $<\sim 10^{-10}$ )  
⇒ auf  $\sim 1$  normieren, wenn der Fit nicht passt

# gnuplot : Einfache 3D-Plots

- einfachster 3D-Plot:  
`plot ... with image`
- x-y-Matrix in 2D, z-Werte farbcodiert.
- Farbbalken als Legende für z-Werte. Angesprochen als `colorbox (cb)`.
- In Datenfiles:  
3 Spalten für x, y, z.  
Matrix muss vollständig und sortiert sein!
- oder `matrix`  
`plot 'file' matrix ...`

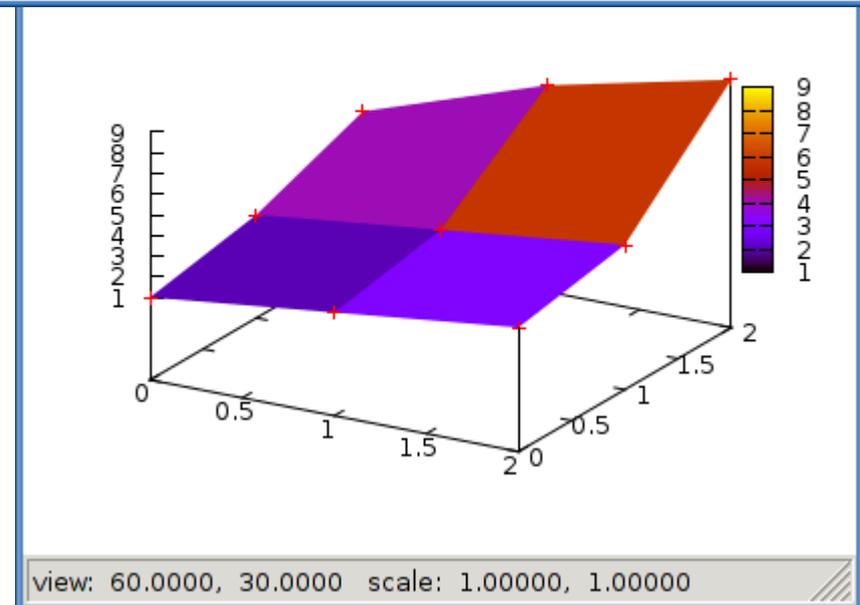
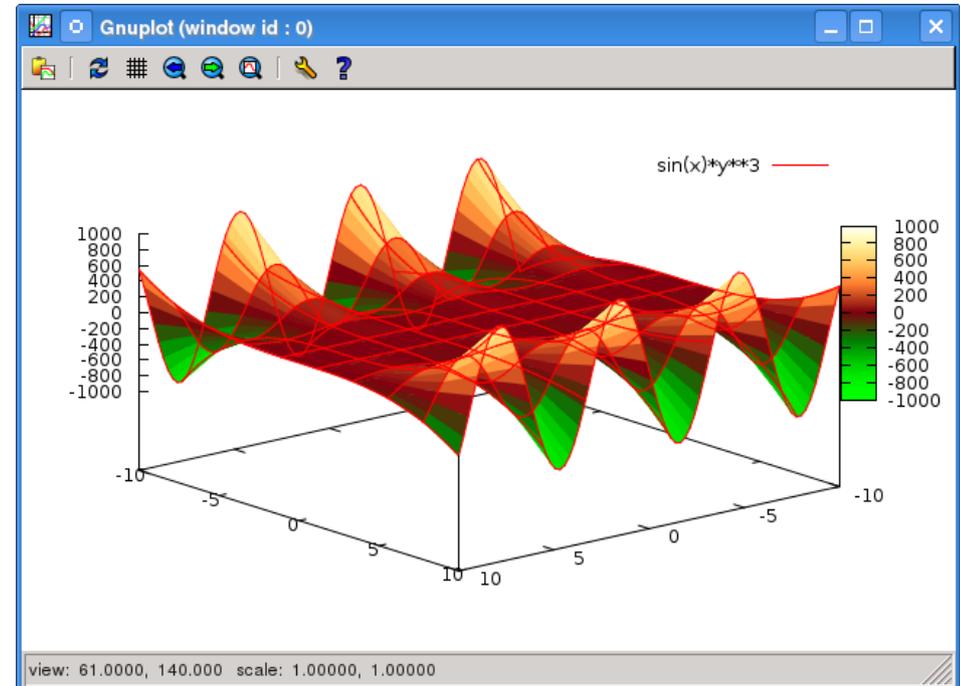
```
1 2 3  
2 3 4  
4 7 9
```

```
# x y z  
1 1 1  
2 1 2  
3 1 3  
  
1 2 2  
2 2 3  
3 2 4  
  
1 3 4  
2 3 7  
3 3 9
```



# gnuplot : echte 3D-Plots

- eigener Befehl für 3D-Plots:  
`splot`
- für „bunte“ Plots:  
`set pm3d`
- unabhängige Variablen:  
`x` und `y`
- In Datenfiles:  
3 Spalten,  
x-Blöcke mit Leerzeilen  
getrennt
- oder  
`matrix`



- parametrischer Plot: Keine Funktion von  $x$ , sondern  $x$  und  $y$  abhängig von einem weiteren Parameter.
- In parametrischen Modus:  
`set parametric`  
⇒  $t$  ist jetzt die unabhängige Variable.
- `plot`-Befehl nimmt zwei Funktionen für  $x(t)$  und  $y(t)$ , mit Komma getrennt:  
`plot [0:2*pi] sin(t),cos(t)`

- Kurve sieht eckig aus? Anzahl Punkte erhöhen:  
`set samples #`
- Kein Taschenrechner zur Hand? gnuplot hilft:  
`gnuplot> print sqrt(24792) + 725 ** 0.274`  
163.532362408873
- Andere Spaltentrenner im Datenfile (z.B. CSV):  
`set datafile separator ","`
- Skript-Ausführung anhalten: `pause seconds`  
`seconds = -1`: auf Benutzer (Enter) warten  
`pause mouse`: Auf Mausklick warten (nicht linke Taste?)
- Alle gnuplot-Einstellungen für einen Plot anschauen, auch die ganzen Default-Einstellungen: Plot darstellen,  
`save 'plot.gnuplot'` und `plot.gnuplot` im Texteditor öffnen.

- Zuerst Ausgabeformat („Terminal“) festlegen. Hier: Postscript  
`set term postscript enhanced color solid`  
Jedes Terminal hat *viele* Optionen ⇒ `help`
- Dann Ausgabedatei öffnen:  
`set out "Datei.ps"`
- Plot ausgeben  
`replot`
- Datei schließen(!)  
`set out`  
Ist die Ausgabedatei nicht lesbar, fehlt meistens „set out“.

- epslatex Terminal: 2 Ausgabedateien:
  - Grafische Elemente als EPS,
  - Text als TeX.
- In TeX wird die .tex-Datei eingebunden, die .eps-Datei wird dann automatisch geladen.
- Für pdflatex: .eps mit epstopdf nach .pdf wandeln.
- (Zur Zeit noch? <sup>1</sup>) Probleme mit der Breite von LaTeX-Text  
⇒ viel Handarbeit nötig: Breite der Legende, Position der Achsenbeschriftungen, etc.

<sup>1</sup> <https://sourceforge.net/p/gnuplot/feature-requests/309/>

- Üblicherweise werden gnuplot-Befehle aus Skript-Dateien ausgeführt.
  - Analysen müssen oft wiederholt mit verschiedenen Daten durchgeführt werden. Die benötigten Einstellungen und Funktionen werden dann nur 1x geschrieben und wiederverwendet.
- Dazu werden einfach die Befehle untereinander in eine Textdatei geschrieben.
- Lange Zeilen können mit \ an jeder Stelle umgebrochen werden.
- Zum Ausführen:
  - gnuplot auf der Kommandozeile den Namen der Datei übergeben. Springt sofort zurück auf die Kommandozeile nach Beendigung des Skripts, außer gnuplot wird mit -persist aufgerufen.
  - gnuplot starten und load "Datei.gnuplot" eingeben. Liest die Datei in der aktuellen Sitzung ein.

Thank you!

# Histogramm aus Datenfile erstellen (einfacher Fall)

- Gegeben: Datei mit ganzzahligen Messwerten, 1 Messung pro Zeile. 532  
514  
529
- Gesucht: Verteilung der Messwerte als Histogramm. 519
- `sort -n datei.dat | uniq -c > out.dat` 530  
`plot 'out.dat' u 2:1 w boxes` 519  
508  
(Hat im Detail ein paar Probleme...) 520  
514
- Für kompliziertere Fälle: awk kann rechnen...
- Oder matlab/octave

